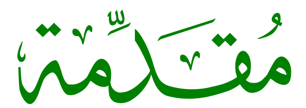


WHAT PROFESSIONALS DON'T KNOW ABOUT ADO





بسم الله الذي لا يضر مع اسمه شيء في الأرض ولا في السماء وهو السميع العليم

بسم الله الرحمن الرحيم



أرجو ألا ينزعج محترفي صناعة تطبيقات قواعد البيانات باستخدام تقنية ADO إن كنت أسأت لهم في عنوان الكتاب (ما لا يعرفه المحترفون عن ADO) ، ولكني قد تطرقت إلى بعض الطرق والمناهج الفكرية الجديدة أثناء تقديمي وتجميعي لمحتوى الكتاب .. كذلك لا يعبر عنوان الكتاب أنه يحتوي على كل ما يتعلق بـ ADO ، فلو قررت فعلا تناول ADO بشكل موسع فلن يكفيني ما يفوق المائة كتاب فهي تقنية جبارة فعلاً ، ولكن جاء هذا الكتاب لمن لا يعلم الكثير عن ADO وكذلك للمحترفين ولكن الذين يجهلون بعضاً من سراديب الاحتراف ، وكذلك أيضا لتوسيع طريقة التعامل مع كائنات ADO وإعطاؤك الدلائل على أنها أكثر من مجرد أداة ربط بين تطبيق وقاعدة بيانات.

اعذروني أن كنت قد أخطأت بأي شكل ، سواء كان الخطأ إملائي أو نحوي أو خطأ في توصيل المعلومة فربما لم يؤتيني ربي هذه الملَكة كي أصل بالمعلومة لعقل القارئ. إذا كان لديك أية استفسارات بعد قراءتك للكتاب قم بمراسلتي شخصياً على البريد الإلكتروني AhmedNegm@WindowsLive.com أو عن طريق الهاتف ٠٠٢٠١١٩٧٧٧٣٤٤ أو ٠٠٢٠١٢٩٤٤٧٩٤٩ ، وذلك حتى نقف وقفة فعلية مع ADO ومميزاتها الخفية.

تعريف بالكاتب

- **الاسم** :

أحمد محمد عبد العظيم نجم (أحمد نجم)

Mr. Ahmed Negm

- **رقم الموبايل** :

٠٠٢٠١١١٩٧٧٧٢٤٤ / ٠٠٢٠١٢٩٤٤٧٩٤٩

- **البريد الإلكتروني** :

AhmedNegm@WindowsLive.Com
Engr_Negmawy@Yahoo.com

- **مكتبة المواضيع المميزة** :

تحتوي مكتبة مواضيعي المميزة على بعض الدروس والمقالات التي قمت بنشرها على الشبكة في موقع أكاديمية فيجوال بيسك للعرب وكذلك موقع منتدى فيجوال بيسك للعرب.

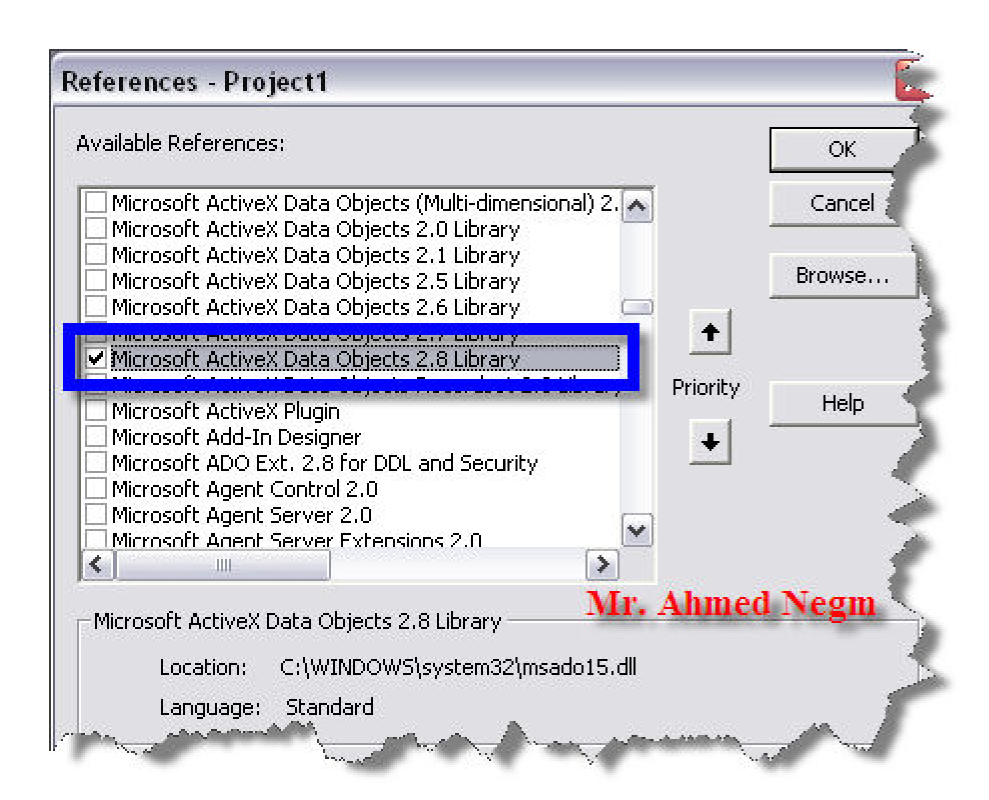
http://www.vba4a.com/vb/showthread.php?t=429

❖ استخدام كائنات ADO ❖

تعرفنا في موضوع سابق على أساسيات قواعد البيانات وكيفية التعامل معها وذلك على الرابط (http://www.vba4a.com/vb/showthread.php?t=15).

وهذا الموضوع هو أول حديثنا عن كائنات بيانات ActiveX Data Objects (ADO) وهي الوسيلة التي من خلالها يتم الوصول إلى جميع أنواع قواعد البيانات من خلال كود Visual Basic.

يمكنك إضافة مراجع ADO للمشروع عن طريق فتحك للقائمة Project ثم أن تختار References ، وقم بتحديد الاختيار الموجود بالصورة :



باختيارك العنصر المحدد وهو (Microsoft ActiveX Data Objects 2.8 Library) ، فقد قمت بتحميل وإضافة هذا المرجع لمشروعك ، وإن لم يظهر لديك

الإصدار ٢،٨ من هذه المكتبة كما هو محدد بالصورة فقم بتحميل حزمة التحديث لـ **Microsoft Visual Studio 6.00** على الرابط التالي (http://www.vba4a.com/vb/showthread.php?t=176) وبذلك يتم حل المشكلة بأسرع ما يمكن إن شاء الله.

نأتي بعد ذلك للتعرف على مكونات أو كائنات **ADO** كما يلي :

- **Connection** : يتيح لك التحكم في الاتصال مع مصدر أو قاعدة البيانات.
- **RecordSet** : يحتوي على السجلات التي تحصل منها على نتائج الاستعلام أو بمعنى أخر فان هذا الكائن هو المسئول عن وصولك للبيانات بالجداول.
- **Command** : يمكنك عن طريق هذا الكائن تنفيذ أوامر واستعلامات قاعدة البيانات.
- **Parameter** : يعمل مع الكائن Command لتعيين معامل داخل استعلام **Query** أو إجراء مخزن **Stored Procedure**.
- **Stream** : يمكنك الاستفادة من الكائن أو الفئة Stream كما هو موضح بالمثال على الرابط التالي (http://www.vba4a.com/vb/showthread.php?t=50).

:: **ملحوظة هامة** ::

يحتوي المرجع (Microsoft ActiveX Data Objects 2.x Library) **على جميع الكائنات التي تم تناولها سابقاً.**

* حيث **x** في العبارة السابقة تعبر عن رقم الإصدار أو التحديث للإصدار.

نبدأ العمل فعلياً مع مكتبة **ADODB** كالآتي :

❖❖ كائن الاتصال Connection ❖❖

أولاً .. لابد من عمل اتصال أو إنشاء قناة اتصال بين التطبيق الخاص بك وبين قاعدة البيانات ، وهذه النوعية من العمليات يتم إنشائها عن طريق استخدام الكائن أو الفئة Connection ، ويكون كود التصريح عن هذه الفئة أو أخذ نسخة من هذه الفئة لاستخدامها في إنشاء الاتصال على النحو التالي :

```
Dim CN As New ADODB.Connection
```

❖❖ يمكنك متابعة الرابط التالي لشركة ***Microsoft*** لزيادة معلوماتك حول **ADO** :

http://support.microsoft.com/default.aspx/kb/183606

نرجع حيث كنا ونقول كي تستطيع العمل مع البيانات الموجودة داخل قاعدة البيانات ، يجب أن تقوم أولا بإنشاء الاتصال مع قاعدة البيانات وذلك من خلال الكائن Connection الذي يساعدك في الاتصال بقاعدة البيانات أو الانفصال عنها. والخطوة الأولى في إنشاء الاتصال هي إنشاء حالة جديدة من هذا الكائن كما في الكود السابق وسأعرضه هذه المرة بأسلوب آخر :

```
Dim CN As ADODB.Connection
Set CN = New ADODB.Connection
```

وكل ما تحتاج إليه بعد إنشاء هذه الحالة هو تحديد معاملات الاتصال واستدعاء الوظيفة Open ، حيث يمكنك تحديد معاملات الاتصال بطريقتين ، الأولى بتخصيص هذه المعاملات للخاصية ConnectionString كما في الكود التالي :

```
Dim strProvider As String

'For MS SQL Server
```

```
strProvider = "User ID= <اسم المستخدم>; Password = < كلمة
السر>; Database = <اسم قاعدة البيانات على السيرفر> ;
Server = <اسم السيرفر أو الآي بي أو الدومين نيم>; Provider =
SQLOLEDB"

'//////////////////////\\\\\\\\\\\\\\\\\\\\

'For MS ACCESS
strProvider = "Provider = Microsoft.Jet.OLEDB.4.0 ; Data
Source = <اسم قاعدة البيانات>"

'=========================================

'Open Connection
CN.ConnectionString = strProvider
CN.Open
```

في هذا الكود قمنا أولا بتخصيص معاملات الاتصال إلى سلسلة من البيانات وهي strProvider ثم تخصيص هذه السلسلة إلى الخاصية Connection String وأخيرا استدعاء الوظيفة Open.

أما الطريقة الثانية لتحديد معاملات الاتصال فتكون بتخصيص أو تمرير هذه المعاملات مباشرة إلى الوظيفة Open التابعة للكائن CN كما يلي :

```
CN.Open "UID= <اسم المستخدم> ; PWD = <كلمة السر> ;
Database = <اسم قاعدة البيانات على السيرفر> ; Server =
<اسم الخادم المستضيف لقاعدة البيانات> ; Driver = ( SQL
SERVER )"
```

:: ملحوظة ::

تحتوي بعض كائنات ADO الأخرى مثل الكائن Command على الخاصية Connection التي تقبل إما سلسلة بيانات الاتصال أو الكائن ADODB.Connection مباشرة.

عند استبدالك للمعاملات المذكورة في جمل الاتصال مثل < اسم المستخدم > يجب حذف هذه العلامات (< وكذلك >) ، ولكن وضعتها فقط للتوضيح ليس إلا.

باتباع إحدى الطريقتين السابقتين تكون قد حققت الاتصال بقاعدة البيانات المذكورة ، وبعد الانتهاء من استخدام الاتصال يمكنك الانفصال عن قاعدة البيانات من خلال الوظيفة Close كما يلي :

```
CN.Close
```

❖❖ مفهوم معاملات الاتصال ❖❖

كما رأينا سوياً فإن معاملات الاتصال (أو سلسلة الاتصال) تحتوي على العديد من المعلومات والتي تشتمل على معرف المستخدم User ID وكلمة المرور Password واسم قاعدة البيانات Database Name حيث يتم كتابة هذه المعاملات في صورة أزواج من الأسماء والقيم مفصولة بعلامة الفاصلة المنقوطة " ; " وعند إنشاء معاملات الاتصال تحتاج إلى تعيين المعلومات التالية :

- نوع قاعدة البيانات التي ترغب في الاتصال بها مثل SQL SERVER أو ACCESS أو ORACLE أو My SQL ... الخ. ويمكنك إجراء الاتصال بواسطة ADO تعيين اسم مشغل ODBC من خلال المعامل Driver أو تعيين اسم مزود OLEDB من خلال قيمة المعامل Provider.
- بيانات عملية الدخول لقاعدة البيانات (Sign-In) إن وجدت والتي تشتمل على قيم اسم المستخدم User ID أو UID وكلمة المرور Password أو PWD.
- موقع قاعدة البيانات والذي قد يشتمل على قيمة للخادم Server قيمة لقاعدة البيانات Database وذلك في حالة SQL Server أو اسم قاعدة البيانات فقط في حالة قاعدة بيانات MS ACCESS.

الكود التالي يوضح بعض الأمثلة على معاملات اتصال ADO :

```
'For MS ACCESS
```

```
"Provider = Microsoft.Jet.OLEDB.4.0 ; Data Source =
C:\Books.mdb"

'or

"Driver = Microsoft Access Driver (*/mdb); DBQ =
C:\Books.mdb"

'===============================================

'For MS SQL SERVER
"Provider = SQLOLEDB ; Password = 0119777244 ; User ID =
Negm; Server = MyServer ; Database = Books"

'or

"UID = Negm ; PWD = 0119777244 ; Database = Books ;
Server = MyServer ; Driver = ( SQL Server )"
```

❖ وعن هذا الكود ، نوضح ما يلي :

- في المثال الأول تم تعيين معاملات الاتصال بقاعدة البيانات **Ms Access** باستخدام مزود OLEDB.
- في المثال الثاني تم تعيين معاملات الاتصال بقاعدة البيانات **Ms Access** باستخدام مشغل ODBC.
- في المثال الثالث تم تعيين معاملات الاتصال بقاعدة البيانات **SQL SERVER** باستخدام مزود OLEDB.
- في المثال الأخير تم تعيين معاملات الاتصال بقاعدة البيانات **SQL *SERVER*** باستخدام مشغل SQL ODBC.

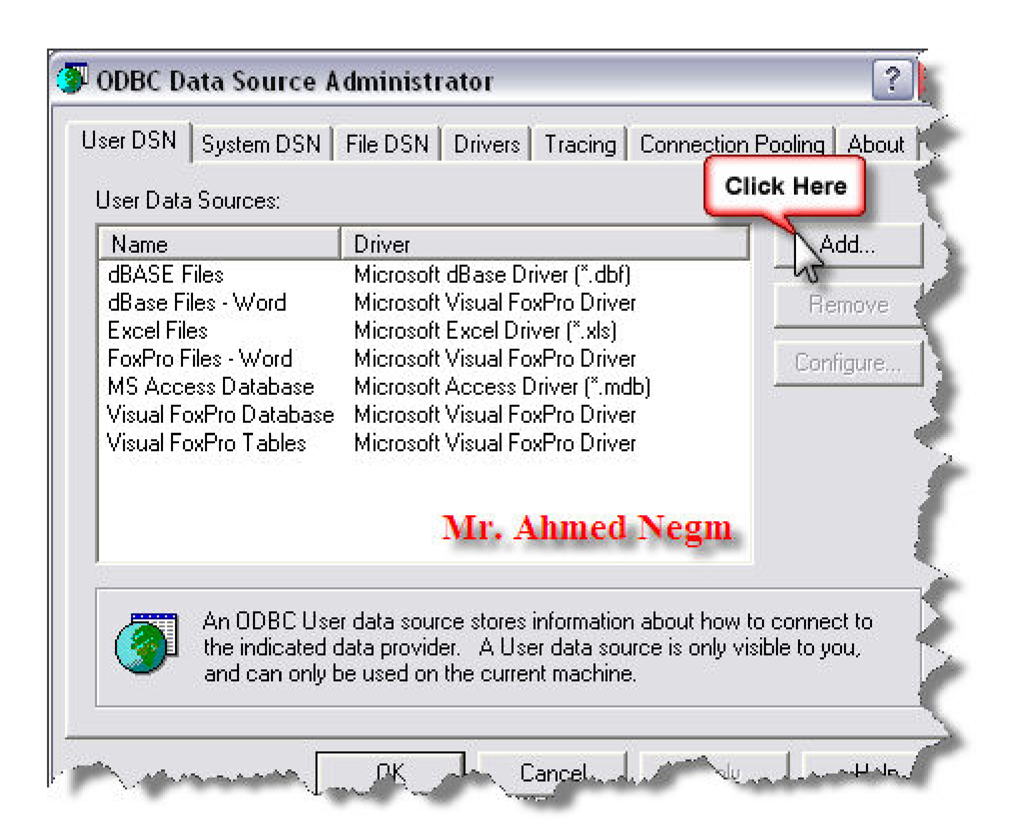
أحد اختياراتك الهامة عند إنشائك لمعامل الاتصال هو استخدام مشغل ODBC أو مزود OLE DB وذلك بتعيين قيمة للمعامل Driver أو المعامل Provider وليس لكلاهما معاً ، حيث تعتبر ODBC اختصاراً للعبارة **Open Database Connectivity** وهي تقنية قياسية لمشغلات قواعد البيانات الموجودة منذ سنوات. أما OLE DB فهي أحدث من سابقتها ، لذا يوصى باستخدامها دائماً قدر الإمكان.

❖❖ استخدام اسم مصدر البيانات ❖❖

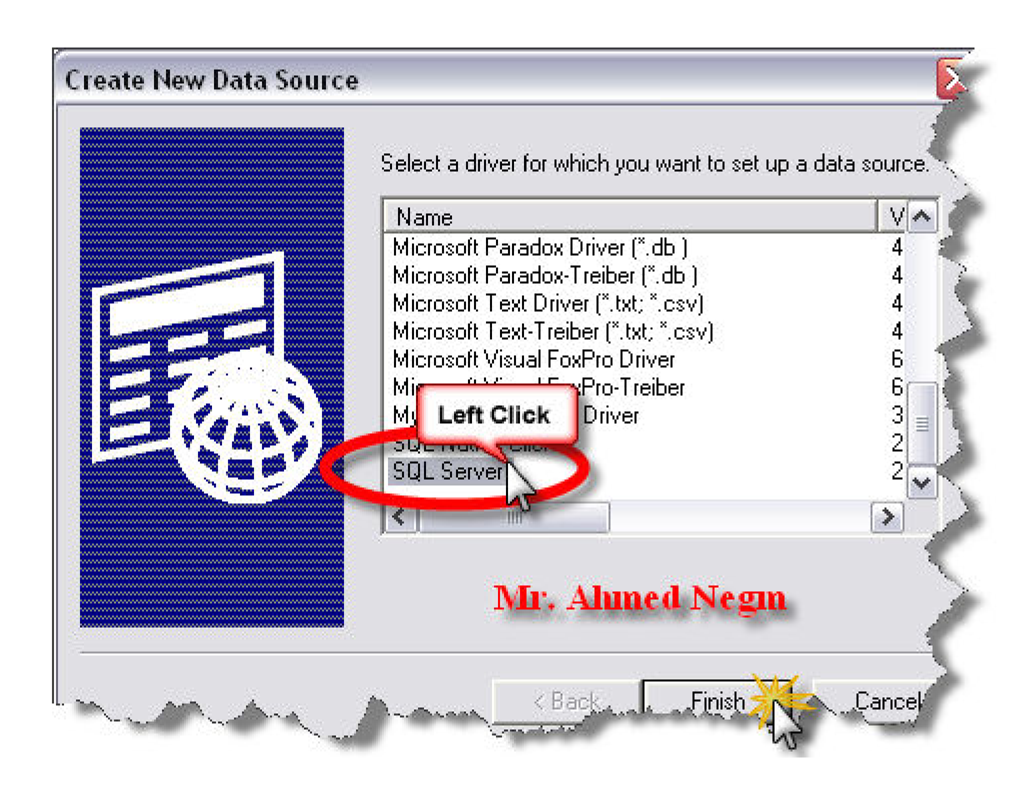
على الرغم من عدم استخدامها بكثرة كما كانت من قبل ، يمكنك استخدام اسم مصدر البيانات (DSN) Data Source Name ضمن سلسلة الاتصال لتعيين مصدر بيانات ODBC كما في الكود التالي :

```
CN.Open "DSN = LocalServer ; UID = Negm ; PWD =
0119777244"
```

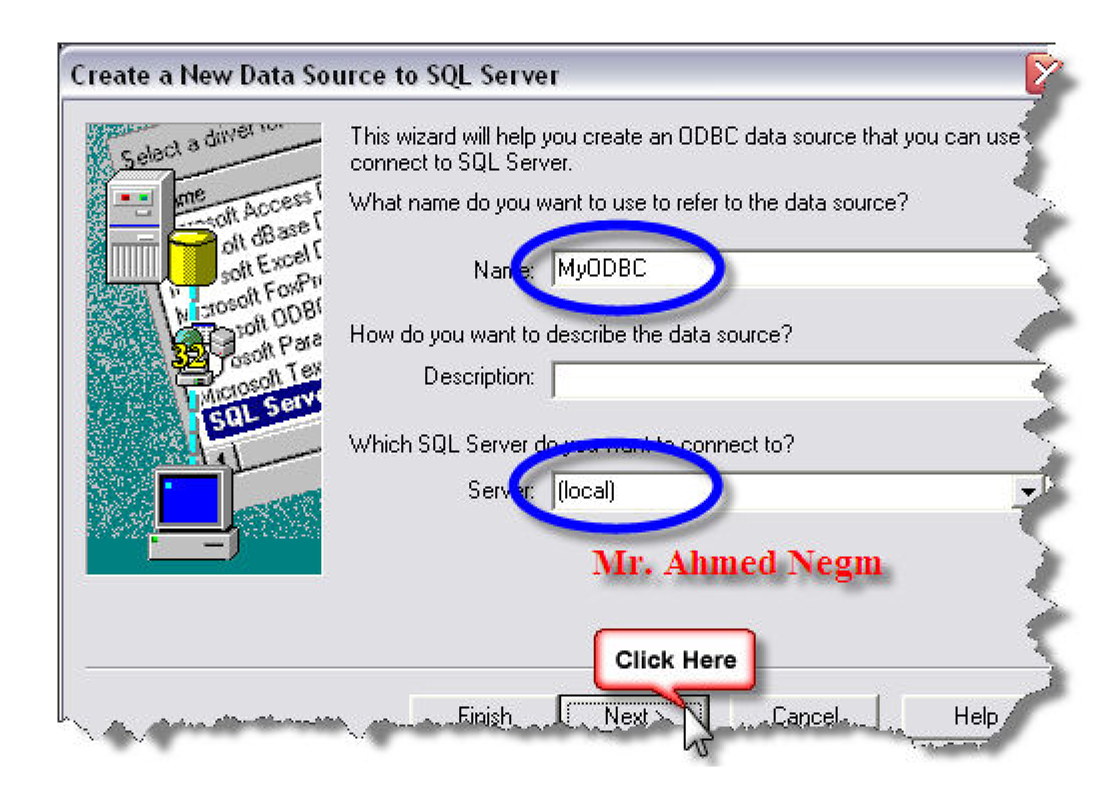
وكما ترى فان سلسلة الاتصال في هذا الكود بسيطة للغاية. فهي تحتوي فقط على DSN واسم المستخدم وكلمة المرور حيث تم تخزين بقية البيانات على مصدر بيانات ODBC المعرف على الحاسب. لإعداد مصدر بيانات ODBC على حاسبك ، افتح معي لوحة التحكم **Control Panel** ثم اضغط على **Administrative Tools** ثم اختر **Data Sources** وبعد ذلك ستظهر أمامك نافذة **ODBC Data Source Administrator** التي يمكنك استخدامها في إدخال بيانات الاتصال حيث يتم تخصيص **System DSN** لجميع المستخدمين بينما يختص **User DSN** بالمستخدم الحالي فقط ... انظر الصور التالية لإنشاء اسم اتصال ODBC خاص بك :



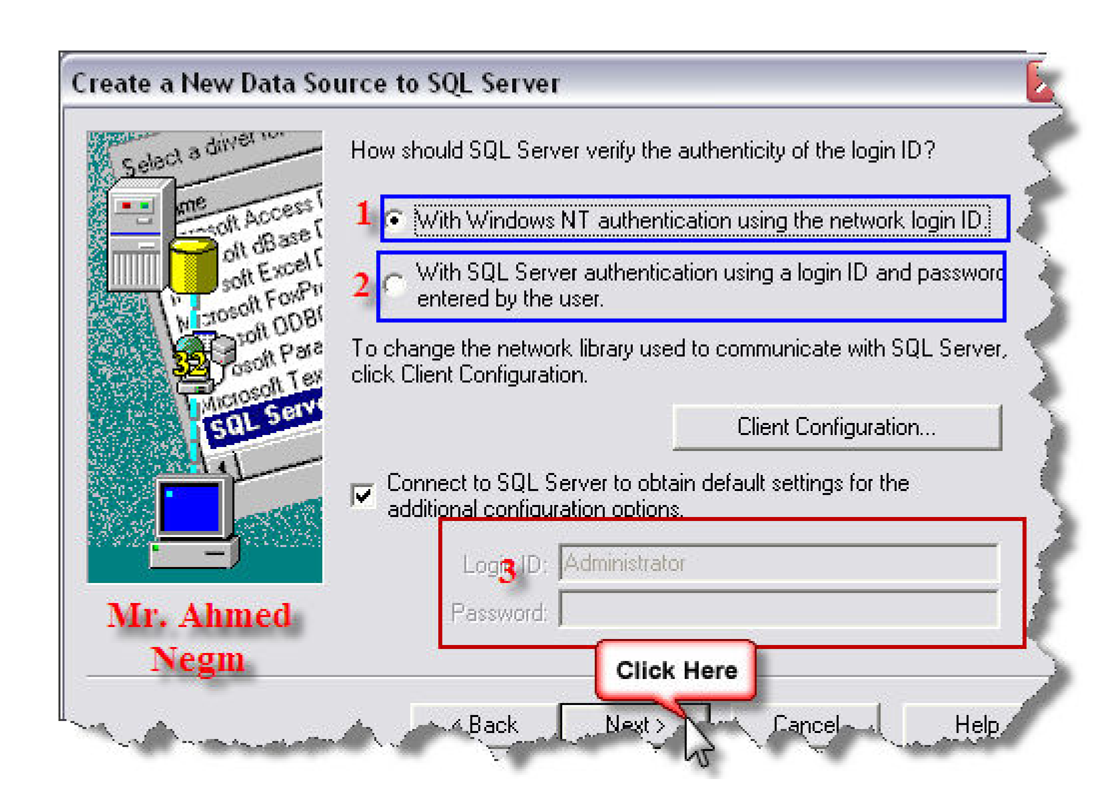
في الصورة التالية قم بتحديد نوع محرك قاعدة البيانات ، وهنا في المثال قمنا بتحديده على أنه SQL Server كما في الصورة أدناه



في الصورة التالية ، قمنا بتعيين ODBC Name ، وهو اسم المشغل الخاص بك والذي سيظهر فيما بعد ضمن قائمة مشغلات ODBC ، وكذلك بيانات الخادم (السيرفر) المستضيف لقاعدة البيانات.



في الصورة التالية قمنا بتحديد بيانات الاتصال أو الدخول للخادم ، ولكن لاحظ الآتي : بما أننا نعمل مع MS SQL Server فهنا لدينا خيارين : الأول ، وهو استخدام Windows NT Authentication ولا داع لاستخدام اسم مستخدم أو كلمة مرور ولهذا يمكنك اختيار الاختيار رقم (١) كما هو بالصورة ... أما الخيار الثاني فنحن بصدد استخدم SQL Server Authentication وعندئذ سيتطلب منا إدخال اسم المستخدم بالإضافة لكلمة المرور كما هو بالجزء رقم (٣) من الصورة.



اختر اسم قاعدة البيانات كما هو موضح بالصورة التالية.



بعد الضغط على مفتاح **Next** كما في الصورة السابقة ستظهر نافذة أخرى ، قم بالضغط على مفتاح **Finish** دون إحداث أية تغيرات ثم سيظهر تقرير بما تم إنشاؤه فاضغط فيه على المفتاح **OK** وستظهر هذه النافذة :



قم بمقارنة الصورة السابقة بالصورة الثانية في هذا الموضوع ، وستلاحظ ما الفرق.

❖❖ استخدام الوظيفة Execute ❖❖

يحتوي الكائن Connection على الوظيفة Execute التي تستخدم في تشغيل عبارة SQL على مصدر البيانات. وإذا قامت هذه العبارة باسترجاع سجلات من مصدر البيانات ، يمكنك استخدام هذه السجلات ببساطة شديدة عن طريق تخصيص القيمة المرتجعة من الوظيفة Execute إلى كائن مجموعة سجلات RecordSet Object.

:: ملحوظة ::

عند العمل مع ADO ، يمكنك الوصول إلى نفس الهدف من خلال أكثر من طريقة. ومثال ذلك عملية استرجاع البيانات التي يمكن تنفيذها عن طريق أي من الكائنات RecordSet و Command و Connection حيث يحتوي كل منهم على وظيفة لاسترجاع البيانات من قاعدة البيانات.

لتوضيح كيفية استخدام الوظيفة Execute في استرجاعه البيانات ، قم أولاً بإنشاء قاعدة بيانات من خلال برنامج MS ACCESS بالاسم **Students.mdb** وأضف لها جدول واحد بنفس الاسم على أن يحتوي هذا الجدول على ثلاثة حقول St_ID و St_FName و St_LName ، وتابع الخطوات التالية :

- افتح بيئة تطوير **Visual Basic 6.00** ، ثم قم بإنشاء مشروع قياسي عادي **Standard EXE**.
- قم بإضافة مرجع لكائنات **ADO** ، كما تم التوضيح بالصورة رقم (١).
- قم بإضافة مربع سرد (**ListBox**) من صندوق الأدوات وأعد تسميته مثلا إلى **lstStudents**.
- قم بإضافة مفتاح أمر (**Command Button**) من صندوق الأدوات وأعد تسميته إلى **cmdLoadList** وقم بتغيير عنوانه (**Caption**) إلى **Run Query**.
- قم بإضافة الكود التالي إلى الحدث **Click** الخاص بمفتاح الأمر السابق :

```
Dim strConnect As String
Dim CN As New ADODB.Connection
Dim RS As New ADODB.Recordset

CN.Open "Provider = Microsoft.Jet.OLEDB.4.0 ; Data Source
= Students.mdb"

Set RS = CN.Execute("SELECT * FROM STUDENTS")

Do While Not RS.EOF
    lstStudents.AddItem RS!ST_FNAME & " " & RS!ST_LNAME
```

```
    RS.MoveNext
Loop

RS.Close
CN.Close

Set RS = Nothing
Set CN = Nothing
```

لتفاصيل المثال ... يمكنك تحميله من على الرابط التالي :

(http://www.vb4arab.com/vb/uploaded/13807/01249069849.rar)

في الكود السابق تم استدعاء الوظيفة Execute التي تنتمي للكائن Connection حيث قامت بدورها بإرجاع مجموعة سجلات RecordSet التي تم تخزينها بالمتغير RS. وبعد ذلك تم استخدام الدوارة أو الحلقة التكرارية المسماة DO .. LOOP للتحرك خلال مجموعة السجلات وإضافة اسم كل طالب ورقمه إلى مرع السرد أو ما يعرف على أنه ListBox.

يمكنك أيضاً استخدام الوظيفة Execute لتنفيذ عبارات SQL التي لا تقوم بإرجاع مجموعة سجلات كحذف السجلات أو إضافتها كما في الكود التالي الذي يقوم بحذف أحد السجلات :

```
CN.Execute "DELETE FROM Students WHERE st_FName = 'Asmaa'
AND st_LName = 'Mohammed'"
```

❖❖ العمليات الأساسية لمجموعة السجلات ❖❖
❖❖ RecordSet ❖❖

تعرفنا فيما سبق على كيفية الحصول على البيانات من قاعدة البيانات ووضعها داخل كائن مجموعة السجلات RecordSet التي تحتوي بدورها على صفوف من البيانات الموجودة داخل جدول أو أكثر من جداول قاعدة البيانات وذلك من خلال تنفيذ أحد الاستعلامات. وعلى الرغم من أن الاستعلام ربما يشتمل على اكثر من جدول إلا أن مجموعة السجلات الناتجة تبدو بالنسبة للبرنامج وكأنها جدول واحد فقط يحتوي على أسماء الحقول وقيمها المختلفة ، حيث تشكل كل مجموعة من الحقول سجل واحد ، كما تتجمع السجلات كلها لتكوين مجموعة السجلات الناتجة.

سنقوم فيما يلي بالتعرف على كيفية إنشاء مجموعة سجلات RecordSet وعرض سجلاتها من خلال كود **Visual Basic**.

❖❖ إنشاء مجموعة السجلات باستخدام الاستعلامات ❖❖

تعلمنا فيما سبق كيفية إنشاء وملء الكائن RecordSet باستخدام ConnectionObject.Execute إلا أن هذا الكائن يحتوي على مجموعة من الوظائف والخصائص المستخدمة في استرجاع البيانات. وكما هو الحال مع جميع الكائنات يجب أن تقوم أولا بإنشاء حالة جديدة من الكائن RecordSet قبل أن تقوم باستخدامه كما في الكود التالي :

```
Dim rsStudent As New ADODB.Recordset
```

وبعد ذلك يمكنك استخدام خصائص هذا الكائن لتعريف الاتصال ومصدر السجلات ونوع مجموعة السجلات. لتعيين مصدر بيانات الكائن Recordset ، قم بتخصيص

كائن الاتصال أو سلسلة الاتصال إلى الخاصية ActiveConnection كما يلي:

```
rsStudent.ActiveConnection = CN
'OR
rsStudent.ActiveConnection = "DSN = Students"
```

حيث:

- يفترض في السطر الأول من الكود أن CN تمثل اتصال مفتوح يشير إلى مصدر البيانات كما أوضحنا من قبل.
- في السطر الثاني تم استخدام سلسلة اتصال وفي هذه الحالة يتم إنشاء كائن الاتصال ضمنياً.

يوضح الكود التالي استخدام الوظيفة RecordSet.Open لملء مجموعة السجلات RecordSet بالبيانات حيث يتم أولا إنشاء كائن RecordSet جديد ثم إضافة بياناته إلى مربع السرد (**ListBox**) الموجود بالنموذج. قم بإضافة نموذج جديد للتطبيق الحالي وبنفس مواصفات النموذج السابق ثم قم بإدخال الكود التالي في إجراء الحدث Click للمفتاح الموجود بالنموذج كما سبق:

```
Dim strConnect As String
Dim strSQL As String
Dim CN As New ADODB.Connection
Dim RS As New ADODB.Recordset

strConnect = "Provider = Microsoft.Jet.OLEDB.4.0 ; Data
Source = Students.mdb"
strSQL = "SELECT * FROM Students ORDER BY ST_ID"

CN.Open strConnect

RS.Open strSQL, CN, adOpenStatic, adLockReadOnly

Do While Not RS.EOF
    lstStudents.AddItem RS!ST_ID & " :  " & RS!ST_FNAME &
" " & RS!ST_LNAME
    RS.MoveNext
Loop

RS.Close
```

```
CN.Close

Set RS = Nothing
Set CN = Nothing
```

يمكنك تحميل المثال من الرابط التالي :

(http://www.vb4arab.com/vb/uploaded/13807/01249070053.rar)

بعد تشغيلك للبرنامج أو التطبيق الذي قمنا بكتابة كوده السابق ، ستكون النتيجة كما يلي في الصورة :



❖❖ إظهار محتويات الحقول ❖❖

استخدمنا في الكود السابق الأدلة في الحصول على بيانات الحقول من مجموعة حقول الكائن RecordSet حيث قامت عبارة SQL بإرجاع ثلاثة حقول هي كود الطالب والاسم الأول والاسم الأخير وتأخذ الدليل أو الترتيب ٠ والدليل ١ والدليل ٢ على الترتيب. ولكن يمكنك استخدام أسماء الحقول بدلا من الأدلة كما فعلنا مع الكائن

Connection. وعلى ذلك يمكنك استرجاع قيمة الحقل الأول في الكود السابق باستخدام أي من الصيغتين التاليتين :

```
RS.Fields("st_ID")
'OR
RS.Fields(0)
```

❖❖ التنقل داخل مجموعة السجلات ❖❖

بعد أن تقوم بإرجاع البيانات من مصدر البيانات إلى مجموعة السجلات RecordSet يمكنك الوصول إلى قيم الحقول الموجودة بالسجل الحالي وتعديلها كما رأينا في الأمثلة السابقة. ويمكنك تصور مجموعة السجلات كما لو كانت ملف طويل ، وفي أي وقت يكون السجل الحالي عبارة عن مؤشر لمكان داخل هذا الملف. وكي نستطيع العمل مع السجلات المختلفة الموجودة داخل مجموعة السجلات الناتجة ، يمكنك استخدام وظائف الانتقال التالية لتغيير السجل الحالي :

- الوظيفة **MoveFirst** : وتتسبب في الانتقال إلى السجل الأول بمجموعة السجلات بعد علامة بداية الملف (BOF) مباشرةً. يشير اختصار (BOF) إلى **Beginning Of File**.
- الوظيفة **MoveLast** : وتتسبب في الانتقال إلى السجل الأخير بمجموعة السجلات قبل علامة نهاية الملف (EOF) مباشرةً. يشير اختصار (EOF) إلى **End Of File**.
- الوظيفة **MoveNext** : وتتسبب في الانتقال إلى السجل التالي بمجموعة السجلات ناحية علامة الملف EOF.
- الوظيفة **MovePrevious** : وتتسبب في الانتقال إلى السجل السابق بمجموعة السجلات ناحية علامة بداية الملف BOF.
- الوظيفة **Move** : وتتسبب في الانتقال للأمام أو للخلف عدداً معيناً من السجلات.

:: **ملحوظة** ::

تعبر كل من BOF و EOF على الخصائص التي توضح نقاط بداية ونهاية مجموعة السجلات على الترتيب.

❖❖ تعيين نوع مؤشر السجل الحالي ❖❖

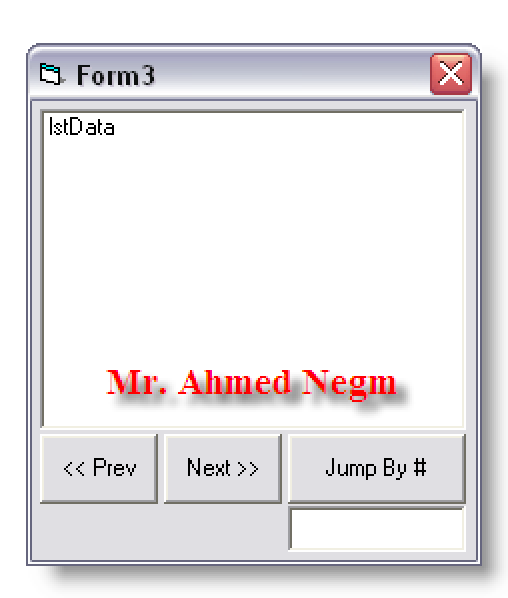
قمنا في الأمثلة السابقة باستخدام وظيفتين من وظائف التنقل بين السجلات وهما الوظيفة MoveFirst والوظيفة MoveNext كما استخدمنا الحلقة التكرارية Do While … Loop للتحرك إلى الأمام داخل مجموعة سجلات مفتوحة حتى نصل إلى نهاية هذه المجموعة ودلالة ذلك ظهور القيمة True داخل الخاصية EOF. ولكن إذا قمنا في الكود السابق باستخدام الوظيفة MovePrevious أو MoveLast فسيقوم المترجم **Compiler** بالاعتراض على الفور وإظهار رسالة الخطأ المناسبة وذلك لعدم صحة نوع المحث Cursor Type المستخدم والذي يشير لدوره إلى المكان الحالي كما في نافذة محث الأوامر (**DOS**) تماماً. ولأننا لم نقم بتعيين قيمة الخاصية Recordset.CursorType، يتم استخدام قيمتها الافتراضية adOpenForwardOnly والتي لا تدعم الوظيفة MovePrevious أو MoveLast.

سنقوم فيما يلي بتوضيح استخدام وظائف التنقل بين السجلات من خلال مثال عملي بسيط .. تابع معي الخطوات التالية :

- قم بإضافة نموذج جديد إلى التطبيق الحالي مع جعله نموذج بدء التشغيل.
- قم بإضافة مربع السرد (**ListBox**) من صندوق الأدوات.
- قم بإضافة ثلاثة مفاتيح أمر (**3 Command Buttons**) أيضاً من صندوق الأدوات.

- قم بإضافة مربع نص (TextBox) من مربع الأدوات إلى النموذج.
- قم بتغيير اسم الـ ListBox إلى lstData
- قم بتغيير اسم المفتاح على اليسار إلى cmdPrev
- والمفتاح الأوسط إلى الاسم cmdNext
- والمفتاح الأيمن cmdJump
- ومربع النص إلى txtJumb

حتى يظهر لديك التصميم بهذا الشكل :



بعد تصميم الشاشة السابقة واتباع تعليمات تغيير أسماء الكائنات على النموذج قم بكتابة الكود التالي :

```
Option Explicit

Dim RS As New ADODB.Recordset

Dim strConnect As String
Dim strSQL As String

'--------------------------------

Private Sub cmdJump_Click()

If Int(txtJumb.Text) > RS.RecordCount - 1 Then
    MsgBox "Sorry !! it is an invalid Position"
```

```
    txtJumb.Text = ""
    txtJumb.SetFocus
    Exit Sub
End If

RS.Move Int(txtJumb.Text), 1
Call Display_Current_Record

End Sub

'--------------------------------

Private Sub cmdNext_Click()

RS.MoveNext
Call Display_Current_Record

End Sub

Private Sub cmdPrev_Click()

RS.MovePrevious
Call Display_Current_Record

End Sub

'--------------------------------

Private Sub Form_Load()

strConnect = "Provider = Microsoft.Jet.OLEDB.4.0 ; Data
Source = Students.mdb"
strSQL = "SELECT * FROM Students"

RS.CursorType = adOpenStatic

RS.Open strSQL, strConnect

Call Display_Current_Record

End Sub

'--------------------------------

Private Sub Display_Current_Record()

Dim i As Integer
Dim s As String

If RS.BOF Then RS.MoveFirst
If RS.EOF Then RS.MoveLast
```

```
lstData.Clear

For i = 0 To RS.Fields.Count - 1

    s = RS.Fields(i).Name & " :  " & RS.Fields(i).Value
    lstData.AddItem s

Next i

Me.Caption = "Current Postion = " & RS.AbsolutePosition

End Sub
```

أو للاستفادة من المثال بشكل مباشر يمكنك تحميله من الرابط :

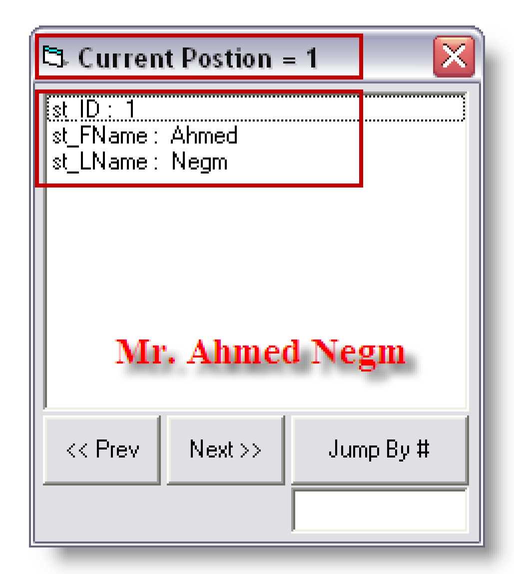
(http://www.vb4arab.com/vb/uploaded/13807/01249081503.rar)

:: **ملحوظة هامة** ::

قمنا فيما سبق بالوصول مباشرة لقاعدة البيانات دون استخدام كائن Connection، وذلك عمداً مني حتى أضع لك أكثر من طريقة للتعامل مع قاعدة البيانات.

قمنا في الكود السابق بإنشاء مجموعة بيانات باستخدام محث ساكن **Static Cursor** وذلك بتخصيص القيمة adOpenStatic للخاصية CursorType والتي تسمح بالتنقل بين السجلات في كلا الاتجاهين على عكس القيمة adForwardOnly التي استخدمناها من قبل والتي تسمح بالتقدم للأمام فقط. وبعد ذلك قمنا بإنشاء الإجراء الفرعي المسمى (Display_Current_Record) التي يتم فيها اختبار الخاصيتين BOF و EOF أولا قبل عرض أسماء وقيم حقول السجل الحالي. وهذه الخطوة ضرورية للغاية وذلك لان محاولة الوصول إلى حقل غير موجود بالسجل الحالي يؤدي إلى حدوث خطأ في عملية الاتصال.

قم بتشغيل التطبيق السابق ولا تنسى أن تجعل النموذج الجديد **Form3** هو نموذج بدء التشغيل، وعند تشغيلك للتطبيق تحصل النتائج كما بالصورة :



❖❖ الحصول على عدد السجلات ❖❖

مـــن الخـــصائص الهامـــة والمـــصاحبة للكـــائن Recordset الخاصـــية RecordCount المستخدمة لاسترجاع عدد سجلات المجموعة. فعلى الرغم من إمكانية استخدام خاصية EOF لحساب عدد السجلات من بداية المجموعة إلى نهايتها، إلا أن استخدام الخاصية RecordCount يوفر العديد من خطوات المعالجة كما في الكود التالي:

```
rsBooks.Open "SELECT * FROM BooksData"

If rsBooks.RecordCount = 0 Then
     MsgBox "No books entered yet !!"
Else
     MsgBox "Number of books registered = " &
rsBooks.RecordCount
End If
```

❖❖ فرز وترشيح مجموعة السجلات ❖❖

يمكنك كما تعلم استخدام عبارة WHERE لترشيح مجموعة السجلات الناتجة من عبارة SQL ، كما يمكنك استخدام عبارة ORDER BY لفرز هذه السجلات وترتيبها ترتيباً معيناً تبعاً لحقل أو اكثر من الحقول الموجودة بقاعدة البيانات.

يحتوي الكائن Recordset على الخاصيتين Filter و Sort التي يمكنها أداء نفس المهام السابقة دون التغيير في استعلام SQL المستخدم ، حيث يمكنك استخدام الخاصية Sort لترتيب السجلات بتخصيصها بقيم الحقول المستخدمة في الترتيب مفصولة بالعلامة " , " كما يلي :

```
RS.Sort = "State , City , FirstName , LastName"
RS.Sort = "Age DESC , FirstName"
```

وكما في عبارة ORDER BY ، يمكنك تعيين إذا ما كان الترتيب تنازلياً أم تصاعدياً من خلال استخدام كلمة DESC مع الترتيب التنازلي. يمكنك أيضا استخدام الخاصية Filter بنفس طريقة استخدام عبارة WHERE كما يلي :

```
RS.Filter = "Age > 20 AND FirstName LIKE 'A%'"
RS.Filter = "State = 'EGYPT' OR State = 'KSA'"
```

:: ملحوظة ::

عند تعيين أي من الخاصيتين Filter أو Sort وإسناد قيم إليهما فانه يتم تحديث محتويات مجموعة السجلات Recordset تلقائياً كي تعكس التغييرات الجديدة.

❖❖ تغيير البيانات داخل مجموعة السجلات ❖❖

بعد أن عرفت كيفية الحصول على البيانات من قاعدة البيانات وكيفية عرضها على النموذج ، من الضروري الآن التعرف على كيفية تغيير أو تحديث هذه البيانات ، فإذا

قمت بإعداد مجموعة السجلات بصورة صحيحة ، يمكنك بسهولة شديدة تغيير البيانات التحتية الموجودة بقاعدة البيانات. كل ما تحتاج إليه هو الانتقال إلى السجل المطلوب ثم تخصيص قيمة جديدة لكل حقل من الحقول التي ترغب في تحديث بياناتها وأخيراً يتم استدعاء الإجراء أو الوظيفة المسئولة عن الحفظ والتحديث في قاعدة البيانات كما بالمثال الآتي:

```
Rs.Fields("FirstName").Value = "Asmaa"
Rs.Update
```

يمكنك كذلك إضافة سجلات جديدة إلى الكائن Recordset باتباع نفس الخطوات السابقة مع خطوة واحدة إضافية كما يلي:

- استدعاء الوظيفة AddNew.
- تخصيص القيم للحقول.
- استدعاء الوظيفة Update.

يوضح الكود التالي كيفية إضافة سجل جديد إلى جدول **Student**:

```
Dim RS As New ADODB.Recordset
Dim strConnect As String, strSQL As String

'Open a recordset

strConnect = "Provider = Microsoft.Jet.OLEDB.4.0 ; Data
Source = Students.mdb"
strSQL = "SELECT * FROM Students"

RS.CursorType = adOpenDynamic
RS.Open strSQL, strConnect

RS.AddNew
    RS!st_ID = 8
    RS!st_FName = "Abdo"
    RS!st_LName = "Negm"
RS.Update

RS.Close
```

```
MsgBox "A new record has been added successfully !!", vbInformation
```

وكما هو الحال كذلك مع عبارة SQL INSERT ، يجب تخصيص القيم للحقول التي لا تستقبل Null قيم فارغة وإلا تظهر رسالة خطأ.

:: ملحوظة ::

إذا لم تقم باستدعاء الوظيفة Update بعد تعيين قيم حقول السجلات الجديدة فلن يتم إضافة هذه السجلات إلى قاعدة البيانات وستذهب بياناتها أدراج الرياح كما يقولون بمجرد الانتقال إلى سجل آخر باستخدام أي من وظائف التنقل بين السجلات التي ذُكرت من قبل.

يمكنك أيضاً حذف السجل الحالي داخل مجموعة السجلات باستخدام الوظيفة Delete كما يلي:

```
Rs.Delete
```

وبمجرد استدعاء هذه الوظيفة (Delete) لا يصبح هناك وجوداً للسجل الحالي ، لذا يجب أن تعقب عملية الحذف هذه باستدعاء إحدى وظائف التنقل بين السجلات قبل محاولة الوصول إلى قيم الحقول.

❖❖ مفهوم تأمين السجلات في ADO ❖❖

تكون مجموعة سجلات ADO افتراضياً للقراءة فقط ، لذا فعند محاولة تخصيص قيمة أحد الحقول في هذه الحالة يتسبب في ظهور رسالة خطأ. فإذا أردت إضافة أو تغيير أو حذف السجلات ، يجب عليك أن تقوم أولا بتعيين قيمة أخرى للخاصية LockType المستخدمة لتحديد نوع تأمين السجلات المستخدم كما في الكود التالي:

```
RS.LockType = adLockOptimistic
```

ولعلك تتساءل الآن عن مدى أهمية هذه الخاصية. والإجابة ببساطة شديدة أنك إذا قمت بتعديل السجلات في قاعدة بيانات متعددة المستخدمين ، يجب أن تكون على دراية كافية بمفهوم تأمين السجلات **Record Locking** والذي يعني منع المستخدمين الآخرين من محاولة التعديل في نفس السجل الموجود في قاعدة البيانات في نفس الوقت. لذا يتم التحكم في تأمين السجلات من خلال الخاصية LockType التي يمكن أن تحتوي على إحدى القيم التالية:

- القيمة (**adLockReadOnly**) : وتقوم بتعيين بيانات مجموعة السجلات للقراءة فقط .

- القيمة (**adLockPessimistic**) : وتوفر التأمين التشاؤمي للسجلات وهو ما يعني تأمين السجلات أثناء تعديله .

- القيمة (**adLockOptimistic**) : وتوفر التأمين التفاؤلي للسجلات وهو ما يعني تأمين السجلات عند استدعاء الوظيفة **Update** فقط .. أي أنك تستطيع تعديل السجلات كما يحلو لك ولكنك لن تستطيع تطبيق هذه التعديلات على قاعدة البيانات حتى ينتهي المستخدم الآخر لنفس مجموعة السجلات من الانتهاء في العمل عليها .

- القيمة (adLockBatchOptimistic) : وتوفر تحديث أكثر من سجل في نفس الوقت من خلال الوظيفة UpdateBatch .

وكذلك أيضاً يجرنا الحديث حول ADO إلى النقطة الخاصة بمشاهدة تغييرات الآخرين وكيفية تحديد مؤشرات الجداول في التعامل مع التطبيقات الشبكية التي تعمل على أكثر من جهاز PC .

❖❖ مشاهدة تغييرات الآخرين ADO ❖❖

عند العمل مع قاعدة بيانات متعددة المستخدمين أي يتم الوصول إليها من خلال شبكة من الحاسبات ، يجب التأكد من دقة البيانات الموجودة داخل مجموعة سجلاتك. فكما ذكرنا سابقاً أن الخاصية CursorType ربما تقيد عملية الانتقال بين السجلات ، وعلى ذلك تعتمد النقطة الرئيسية في تحديد نوع المؤشر المستخدم على كيفية ارتباط مجموعة السجلات بالبيانات الأساسية الموجودة بقاعدة البيانات.

وفيما يلي نوضح القيم المختلفة للخاصية [CursorType] ومدلول كل منها..

- القيمة (adOpenForwardOnly) : تستخدم لتسريع استرجاع البيانات إلى الأمام فقط داخل مجموعة السجلات ... ويمكنك الاستفادة من هذه القيمة في فتح الاتصال بالجدول بهذه الطريقة لتعبئة كائن ComboBox أو ListBox.. الخ ، لأنك لن تكون في حاجة إلى الاحتفاظ بالسجل في الذاكرة بعد قراءته ، وباستخدامك لهذه الطريقة بمجرد قراءة السجل والانتقال إلى السجل التالي له يتم حذف السجلات السابقة جميعها من الذاكرة مما يخفف أعباء العمل في ذاكرة الجهاز .
- القيمة (adOpenKeySet) : تتيح لبرنامجك مشاهدة بعض تغييرات البيانات التي تتم من قبل المستخدمين الآخرين ، ولكنها أدق في حالة العمل على الخادم - على حد علمي .
- القيمة (adOpenDynamic) : تتيح لبرنامجك مشاهدة جميع تغييرات البيانات التي تتم من قبل المستخدمين الآخرين .
- القيمة (adOpenStatic) : لا يمكن من خلال هذه القيمة مشاهدة أي تغييرات من قبل المستخدمين الآخرين ، وذلك لأنك قد استخدمت طريقة الفتح الاستاتيكي (الساكن) للبيانات ،ويمكنك استخدام مثل هذه الطريقة في الفتح مثلاً عند إنشاء وصناعة التقارير .

يعتمد نوع القيم المتاحة لمؤشر السجلات على نوع قاعدة البيانات من قبل جميع قواعد البيانات. ففي حالة قاعدة بيانات **Ms Access** على سبيل المثال ، تكون القيمة الافتراضية للمؤشر هي `adOpenForwardOnly` والتي تستخدم للقراءة السريعة من قاعدة البيانات ، بينما يفضل استخدام المؤشر `adOpenKeySet` في حالة عمليات التحديث والعمليات المركبة الأخرى.

❖❖ كائن الأمر Command ❖❖

عند العمل مع مجموعة من البيانات ، فانك تقضي معظم الوقت في التعامل مع خصائص ووظائف الكائن أو الفئة Recordset. لكن إذا ما أردت استرجاع هذه البيانات ، فستجد أن لا غنى عن استخدام ADO Command الذي يتيح لك تضمين استعلام أو إجراء SQL مخزن داخل كائن قابل للاستخدام اكثر من مرة وبالتالي يكون هو الحل المثالي إذا أردت تنفيذ عملية من العمليات مرات عديدة. ويتم تخزين معاملات الاستعلام أو الإجراء المخزن داخل التجمع Parameters الخاص بالكائن Command. وبعد أن ننتهي من إعداد هذا الكائن يمكنك تغيير معاملاته واستدعاؤه اكثر من مرة.

لتوضيح كيفية استخدام الكائن Command ، انظر معي إلى عبارة SQL التالي التي تقوم باسترجاع جميع الطلبة الذين يبدأ اسمهم الأول بحرف "A" :

```
SELECT * FROM Students WHERE st_FName LIKE 'A%'"
```

سنقوم بتحويل العبارة إلى إجراء مخزن داخل قاعدة بيانات من نوع **Ms SQL Server** كي يقوم المستخدم بتحديد الحرف الأول بنفسه وذلك كما يلي :

```
CREATE PROCEDURE spSearch
@strSearchLetter char(1)
AS
SELECT *
FROM Students
WHERE st_FName LIKE  @strSearchLetter + '%'
```

بعد ذلك يمكنك إنشاء الكائن Command وإخباره بالإجراء المخزن أو ما يدعى **Stored Procedure** والموجود داخل قاعدة بيانات **Ms SQL Server** وتمرير قيمة المعامل المطلوب في ذلك الإجراء كما بالكود التالي :

```
Dim CN As New ADODB.Connection
Dim RS As New ADODB.Recordset
Dim Cmd As New ADODB.Command
```

```
Dim Prm As New ADODB.Parameter


'Open a CONNECTION to the database
CN.Open "Provider = SQLOLEDB ; Password = 0119777244 ;
User ID = Negm ; Server = MyServer ; Database = Students"


'Setup the COMMAND object
Cmd.CommandType = adCmdStoredProc
Cmd.CommandText = "spSearch"
Cmd.ActiveConnection = CN


'Setup PARAMETER object
Prm = Cmd.CreateParameter("@strSearchLetter", adChar,
adParamInput, 1, "A")
Cmd.Parameters.Append (Prm)


'Setup the RECORDSET object
RS.CursorType = adOpenStatic
RS.LockType = adLockOptimistic


'EXECUTE the command
RS.Open Cmd
```

وكما ترى فان هذا الكود يحتوي على جميع كائنات ADO التي شرحناها في هذا الموضوع حتى الآن. كما يحتوي الكائن Command على مجموعة من الكائنات Parameters التي يمثل كل منها معامل داخل الإجراء المخزن. ويمكنك بعد إنشاء الكائن Command احتواء هذه المعاملات واستدعائها مرة أخرى من خلال عدد قليل من اسطر الكود كما يلي:

```
Cmd.Parameters ("@strSearchLetter").Value = "A"
RS.close
RS.Open cmd
```

وفي هذا الكود يتم إعادة فتح مجموعة السجلات مرة أخرى بعد تغيير معاملات الأمر أو الكائن Command حيث يتم استرجاع جميع الأسماء التي تبدأ بالحرف A.

:: **ملحوظة** ::

يحتـوي الكـائن Command **أيـضاً علـى الوظيفـة** Execute **التـي تعمـل بطريقة مشابهة لمثيلتها في الكائن** Connection.

❖❖ مجموعات السجلات المنفصلة ❖❖

تم استحداث مجموعة السجلات المنفصلة أي التي تعمل داخل الذاكرة بمعزل عن قاعدة البيانات المصاحبة ، حيث يمكنك تحديث مجموعة البيانات وحفظها على وحدة تخزين **Storage Device** **إن أحببت وكذلك تحقيق التزامن بينها وبين قاعدة البيانات فيما بعد. وتظهر أهمية السجلات المنفصلة** **Disconnected Recordset** **عند العمل مع البيانات الكبيرة التي يستخدمها آلاف المستخدمين ، ولكن هذه التقنية تقلل من وقت الاتصال وتزيد في سرعة عرض النتائج المطلوبة.**

يمكنك فصل مجموعة السجلات Recordset **عن قاعدة البيانات عن طريق تخصيص القيمة** adUseClient **للخاصية** CursorLocation **التي تحدد مكان المؤشر إذا كان في قاعدة البيانات (جهة الخادم) أو في الذاكرة (جهة العميل). ويمكنك إنشاء مجموعة السجلات المنفصلة باتباع الخطوات الآتية :**

- **قم بإنشاء كائن** Recordset **جديد.**
- **قم بتخصيص القيمة** adUseClient **للخاصية** CursorLocation.
- **قم بملء الكائن** Recordst **بالبيانات من خلال الاتصال بقاعدة البيانات باستخدام الوظيفة** Open **كما سبق وأن شرحنا.**
- **قم بتخصيص القيمة** Nothing **للخاصية** ActiveConnection **حتى يتم تعطيل أو فصل قناة الاتصال بين التطبيق وقاعدة البيانات. وذلك كما بالكود التالي :**

```
Dim CN As New ADODB.Connection
```

```
Dim RS As New ADODB.Recordset

CN.Open "Provider = Microsoft.Jet.OLEDB.4.0 ; Data Source
= Students.mdb"

RS.CursorType = adOpenKeyset
RS.CursorLocation = adUseClient
RS.LockType = adLockOptimistic

RS.Open "SELECT * FROM STUDENTS", CN

RS.ActiveConnection = Nothing
```

❖❖ إنشاء مجموعة السجلات من خلال الكود ❖❖

على الرغم من سهولة ملء مجموعة السجلات Recordset من قاعدة بيانات موجودة بالفعل بسهولة وببساطة شديدة ، إلا انك تستطيع إنشاء مجموعة سجلات من البداية باستخدام الكود عن طريق إنشاء كائن Recordset جديد ثم إنشاء مجموعة الحقول Fields وأخيراً إضافة السجلات باستخدام نفس الوظائف التي ذكرناها سابقاً. تابع الكود التالي كمثال صغير على ما سبق ، وسنقوم بإنشاء مجموعة سجلات تحتوي على حقلين فقط هما Name و Age وذلك بعيدا كل البعد عن أي وجود لقاعدة بيانات :

```
Dim RS As New ADODB.Recordset

'Create Structure
RS.Fields.Append "Name", adVarChar, 25, adFldIsNullable
RS.Fields.Append "Age", adInteger, , adFldMayBeNull

'Add a record
RS.Open

RS.AddNew
    RS.Fields("Name").Value = "Ahmed Negm"
    RS.Fields("Age").Value = 22
    MsgBox  "Data  has  been  saved  successfully  !!",
vbInformation
RS.Update
```

الجديد في هذا الكود هو انك بعيداً كل البعد عن قواعد البيانات ، أنت الآن تعمل بذاكرة جهازك الشخصي .. إذا أدركت هذه النقطة كما هو مطلوب فثق انك قد دخلت مصاف المحترفين ، أعلم أن الفكرة قد تكون صعبة نسبياً حيث قد تعودت دائماً استخدام RS مع كائن CN بالإضافة لقاعدة بيانات ، ولكن هنا الوضع قد اختلف كل الاختلاف فنحن هنا لم نتعرض لقاعدة بيانات من أي نوع على الإطلاق ، فقد قمنا بإنشاء جدول من نوع جديد على الإطلاق وهو جدول وهمي ومكان وجوده هو ذاكرة الجهاز الذي تعمل عليه .. لن أستطيع التحدث عن هذه النقطة أكثر من ذلك ولكن بإمكانك طرح استفسارك على المنتدى في قسم ربط محركات قواعد البيانات بفيجوال بيسك ٦ الملحق بقسم فيجوال بيسك ٦ .. مما ظهر جديدا أيضا في هذا الكود هو استخدام الوظيفة Append التي تقوم بإضافة حقل إلى مجموعة البيانات وتحتوي على أكثر من معامل مثل اسم الحقل ونوع بياناته وطول هذا الحقل وكذلك بعض صفاته مثل عدم استقبال القيمة Null بتعيينك للخاصية adFldIsNullable ، أو إمكانية استقبال قيم فارغة بتعيينك للخاصية adFldMayBeNull ، كما في الحقل Name و Age على الترتيب.

مما جد جديداً هنا ، هو حفظ مجموعة السجلات داخل ملف بتنسيق XML على القرص الصلب باستخدام الكود التالي على سبيل المثال :

```
RS.Save "C:\TempTable.xml", adPersistXML
```

كما يمكنك بكل بساطة استخدام هذا الملف وما حفظ فيه مرة أخرى عن طريق الكود التالي :

```
RS.Open "C:\TempTable.xml"
```

❖❖ تحديث قاعدة البيانات ❖❖

لا أدري من أين ابدأ ، فقد دخلنا في معمعة الـ **ADO** منذ قليل .. ولكن دعونا نبدأ بشكل عادي ولا نابه لما يقابلنا من صعوبة وفقط نستوعب ما هو مكتوب ونقوم

بتطبيقه .. تعرفنا قبل ذلك على كيفية تأمين السجلات لمنع مجموعة المستخدمين من التعامل مع نفس الحقل في نفس الوقت ، وتم مناقشة هذا الموضوع على الرابط التالي : http://www.vba4a.com/vb/showthread.php?t=71 أو كما تم ذكره سابقاً. لكن على الرغم من ذلك يأخذ مفهوم تامين السجلات معنى أخر عند العمل مع مجموعة السجلات المنفصلة. فإذا لم تكن متصلا بقاعدة البيانات ، فأنت بذلك تقوم بالنظر إلى السجلات كما لو كنت تنظر إلى مجموعة من الكتب داخل المكتبة. فإذا قمت بتحديث أو تعديل بيانات مجموعة السجلات ، فلن تنعكس هذه التغييرات على قاعدة البيانات حتى تقوم بتسجيل هذه السجلات.

أشعر بأن عقلك بدأ يضطرب إذا كنت أول مرة تقرأ هذه المعلومات .. تخيل بأنك تم تعيينك كموظف جديد بقسم الأرشيف في إحدى المصالح الحكومية الأكثر غباءً هذا على سبيل المثال وقال لك مديرك عليك تسجيل البيانات الخاصة بالشهر الماضي لأنها لم تسجل بعد ، وقد داهمك الوقت ولم تستطيع إنجاز عملك في المصلحة الحكومية أو أثناء وقت عملك وطلبت من مديرك أخذ العمل والملفات للمنزل كي تنجز أعمالك في هدوء وبدون إزعاج ولا خطأ ، ولكن المشكلة هنا أنك لابد من أخذ هذه الملفات مرة أخرى بعد إنجازها للأرشيف ووضعها في أماكنها مرة أخرى. تعال معي نترجم هذا لكلام برمجي :

يمكنك تحديث قاعدة البيانات بطريقتين ، الأولى بإرجاء الأمر إلى **ADO** كي يقوم بعملية التحديث المجمعة Batch Update ، أما عن الطريقة الثانية فتتم يدويا من خلال الكود بمعالجة كل سجل داخل مجموعة السجلات المنفصلة باستدعاء إجراء مخزن لتحديث سجل قاعدة البيانات المصاحب. وفيما يلي نقوم بإلقاء نظرة خاطفة على كلٍ من الطريقتين.

❖❖ تحديث مجموعة من السجلات ❖❖

يتيح لك التحديث المجمع للسجلات إجراء العديد من التغييرات على مجموعة السجلات وتطبيقها على قاعدة البيانات المصاحبة مرة واحدة من خلال الوظيفة UpdateBatch. ولكي تتمكن من استخدام هذه الوظيفة ، يجب تخصيص القيمة

للخاصية adLockBatchOptimistic الخاصة LockType بمجموعة السجلات كما يلي:

```
RS.LockType = adLockBatchOptimistic
```

وعلى ذلك تشتمل رحلة التحديث على الخطوات الآتية:

- إجراء التغييرات على مجموعة السجلات المنفصلة كما في الكود التالي:

```
RS.Fields("Name").Value = "Ahmed Negm"
RS.Update
' في الإجراء السابق قمنا بالتعديل ، وفي الإجراء التالي سنقوم بعمل حذف لأحد السجلات ، فقط كدليل لك على أنك قد قمت ببعض التغييرات في قاعدة البيانات
RS.MoveNext
RS.Delete
```

- إعادة تنشيط الاتصال مع قاعدة البيانات بتخصيص كائن الاتصال المفتوح للخاصية ActiveConnection كما يلي:

```
RS.ActiveCommand = CN
```

- تحديث قاعدة البيانات من خلال الوظيفة UpdateBatch كما يلي:

```
RS.MarshalOptions = adMarshalModifiedOnly
RS.UpdateBatch
```

وقد استخدمنا الخاصية MarshalOptions التي تحدد إذا ما كنت تريد استرجاع مجموعة السجلات بالكامل أم السجلات التي تم تعديلها فقط. وبذلك تتم عملية التحديث بتغيير قيمة الحقل Name في السجل الأول ثم حذف السجل التالي.

❖❖ تحديث البيانات يدوياً ❖❖

ربما أردت في بعض الأحيان مزيدا من التحكم في عملية التحديث ، وفي هذه الحالة يمكنك استخدام الخاصية Status المصاحبة للكائن Recordset لتحديد نوع التحديث الذي يتم على سجل معين ، وذلك كما في الكود التالي :

```
Do While Not RS.EOF

    Select Case RS.Status
        Case ADODB.RecordStatusEnum.adRecNew
            'INSERT INTO .. هذا سجل جديد ، قم بتنفيذ العبارة
        Case ADODB.RecordStatusEnum.adRecModified
            'UPDATE .. حدث تغيير للسجل ، قم بتنفيذ العبارة
    End Select

RS.MoveNext

Loop

End Sub
```

خاتمة

إلى هنا أكون قد وصلت إلى جولتي الأولى والقصيرة في نفس الوقت مع **ADO** ، راجياً من الله - تعالى - أن أكون قد وفقت في زيادة معرفة القارئ عن **ADO** ولو بشيء يسير .. وانتظرونا في لقاء آخر موسع حول التعامل مع **ADO**.

أكرر مرة أخرى ... إذا تكرمت أخي ووجدت أخطاء فنية في الأكواد أو الروابط المباشرة أو الأمثلة المرفقة أو حتى أخطاء إملائية ، فلا تتردد وراسلني كي نخرج هذا المرجع بأحسن صورة. تابعونا على منتديات أكاديمية فيجوال بيسك للعرب على الرابط **www.vba4a.com** أو منتديات فيجوال بيسك للعرب على الرابط **www.vb4arab.com** وستجدون دوماً ما تطلبونه وتسعون للحصول عليه أثناء مرحلتكم التعليمية بلغات **Visual Basic** ولغات أخرى.

سبحان الله عدد خلقه ورضا نفسه وزنة عرشه ومداد كلماته ... وصلى الله وسلم وبارك على سيد الخلق كلهم سيدنا محمد صلى الله عليه وسلم صلاةً دائمةً إلى يوم الدين.

تحياتي

❖ أحمد نجم ❖

AhmedNegm@windowslive.com
002 011 977 72 44 - 002 012 944 79 49
EGYPT